Target Identification Using Wavelet-based Feature Extraction and Neural Network Classifiers

Jose E. Lopez, Hung Han Chen, Jennifer Saulnier
CYTEL Systems, Inc.
Hudson, MA 01749

ABSTRACT

Classification of combat vehicle types based on acoustic and seismic signals remains a challenging task due to temporal and frequency variability that exists in these passively collected vehicle indicators. This paper presents the results of exploiting the wavelet characteristic of projecting signal dynamics to an efficient temporal/scale (i.e. frequency) decomposition and extracting from that process a set of wavelet-based features for classification using a multilayer feedforward neural network for vehicle classification. This effort is part of a larger project aimed at developing an Integrated Vehicle Classification System Using Wavelet / Neural Network Processing of Acoustic/Seismic Emissions on a Windows PC performed under a Phase II SBIR for the US Army TACOM/ARDEC. The data set used for validation consists of ground combat vehicles (e.g. Tanks (T-62, T-72, M-60), Lightweight Utility Vehicle, Tracked APC and Tank Transporter) recorded at the Aberdeen Test Center, MD. Initial results using wavelet-based feature extraction and a feed-forward neural network vehicle classifier employing the Levenberg-Marquardt deterministic optimization learning scheme will be presented.

1. INTRODUCTION

Acoustic emissions from ground vehicles contains enough information content due to varying machine configurations, transmission, mass and design that sophisticated systems should be able to readily extract this information and separate various vehicle categories leading to a robust, passive surveillance system. A key element in being able to produce a viable advanced pattern recognition system is the ability to develop a robust projection methodology which leads to a low-dimensional signal characterization which can be used to separate the various vehicle targets. A first order metric as to the extensibility and reliability of any identification methodology is it's ability to capture and highlight structure associated with the machinery elements in the vehicle targets while simultaneously mitigating and/or ignoring effects associated with the specific vehicle operational environment.

An important signal decomposition technique, which has proven quite valuable due to it's ability to pick out both long term and transient events in a seamless fashion across the time-frequency plane, is the wavelet transform [1-10]. The viability of using the wavelet transform for the vehicle identification problem will depend on two key elements 1) the ability to develop a wavelet basis function that can be used to highlight unique wavelet structures for each of the vehicle targets and 2) the ability to design an extraction method suitable for real-time applications that can efficiently extract these unique wavelet structures for classification.

The final component in the vehicle identification system will be a sophisticated classification method, which is readily adaptable, extensible and capable of handling complex decision regions in wavelet-based feature space. The emerging role of neural networks to execute on a large variety of pattern recognition problems, especially difficult pattern recognition problems with low signal to noise ratios and non-convex, non-linear separation characteristics provides a powerful solution for problems demanding robust pattern classification performance [11-17]. A strategic blend of these technologies is capable of leading to adaptable advance ground vehicle classification systems.

2. CONTINUOUS WAVELET TRANSFORMS (CWT)

To develop viable wavelet-based monitoring and classification schemes, a means of extracting significant discrimination features from the acoustic signal plays a critical role. Harmonic analysis in the form of a Fourier transform proves problematic for several reasons. First, the transform is global in that localized events in time can effect the entire frequency spectrum. Additionally, the Fourier transform is fundamentally not applicable to real-time monitoring applications due to the mathematical formulation of the transform that operates on the entire time axis. Windowing schemes are thus required to address the real-time feature extraction requirements for capturing

Report Documentation Page

Report Date	Report Type	Dates Covered (from... to)
00 Sep 1999	N/A	-

Title and Subtitle Target Identification Using Wavelet-Based Feature Extraction and Neural Network Classifiers	**Contract Number**
	Grant Number
	Program Element Number
Author(s)	**Project Number**
	Task Number
	Work Unit Number
Performing Organization Name(s) and Address(es) CYTEL Systems, Inc Hudson, MA 01749	**Performing Organization Report Number**
Sponsoring/Monitoring Agency Name(s) and Address(es) Department of the Army, CECOM RDEC Night Vision & Electronic Sensors Directorate AMSEL-RD-NV-D 10221 Burbeck Road Ft. Belvoir, VA 22060-5806	**Sponsor/Monitor's Acronym(s)**
	Sponsor/Monitor's Report Number(s)
Distribution/Availability Statement Approved for public release, distribution unlimited	
Supplementary Notes See also ADM201471, Papers from the Meeting of the MSS Specialty Group on Battlefield Acoustic and Seismic Sensing, Magnetic and Electric Field Sensors (2001) Held in Applied Physics Lab, Johns Hopkins Univ, Laurel, MD. on 24-26 Oct. 2001. Volume 2 (Also includes 1999 and 2000 Meetings), The original document contains color images.	
Abstract	
Subject Terms	
Report Classification unclassified	**Classification of this page** unclassified
Classification of Abstract unclassified	**Limitation of Abstract** UU
Number of Pages 19	

important events localized in time. Unfortunately, fixed windowing schemes imply fixed time-frequency resolution in the time-frequency plane. The problem this poses is the selection of a single window that provides sufficient fidelity discriminating important events in the acoustic signal that are separated by large orders of magnitude along the frequency axis.

The continuous wavelet transform (CWT) resolves the window selection problem with a “zoom-in” and “zoom-out” capability that generates a flexible time-frequency window that automatically narrows (along the time axis) at high center-frequencies and expands (along the time axis) at low center frequencies. The continuous wavelet transform provides this flexible time-frequency analysis by decomposing the acoustic signal over dilated and translated wavelet basis functions.

A wavelet is a function with finite energy, or a member of the function space $L^2(R)$, i.e., a wavelet function satisfies:

$$\int_{-\infty}^{\infty} |\psi(x)|^2 dx < \infty \tag{2.1}$$

In addition, the wavelet function has a zero average or essentially no DC component. A set of basis functions are obtained via dilations and translations of a base wavelet and takes the form:

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}}\,\psi\left(\frac{t-u}{s}\right) \tag{2.2}$$

where u is the translation parameter and s is the dilation parameter. The wavelet transform is then achieved via the inner product of the respective acoustic signal, $f(t)$, with the wavelet basis function of equation (2.2):

$$W_{\psi} f(u,s) = \int_{-\infty}^{\infty} f(t)\frac{1}{\sqrt{s}}\,\psi^*\left(\frac{t-u}{s}\right)dt \tag{2.3}$$

An important interpretation of the wavelet transform is obtained by rewriting equation (2.3) in the form of a convolution product:

$$f(t) * \tilde{\psi}_s(t) = \int_{-\infty}^{\infty} f(\tau)\tilde{\psi}_s(\tau - t)d\tau \tag{2.4}$$

where

$$\tilde{\psi}_s(t) = \frac{1}{\sqrt{s}}\,\psi^*\left(\frac{-t}{s}\right) \tag{2.5}$$

The Fourier transform of Equation (2.5) is $\tilde{\Psi}_s(\omega) = \sqrt{s}\Psi\ (s\omega)$, hence the wavelet transform can be interpreted as convolving the acoustic signal with a family of dilated band-pass filters.

The success of wavelet decomposition resides in maximizing the flexible time-frequency trade-offs available from analyzing acoustic signals with a set of wavelet basis functions. When other approaches are employed, such as time-scale methods based on short-time Fourier methods using Gaussian windows (e.g., short-time Fourier transform (STFT)), selection of a time window automatically determines a fixed frequency window. The time-frequency uncertainty is therefore fixed through out the time-frequency plane. Figure 2.1, left-hand side, illustrates this phenomena.

Each rectangle of Figure 2.1 is representative of the "Heisenberg boxes" that result from fixed time Gaussian windows imposed by the STFT method. The Heisenberg uncertainty principle concerns simultaneous localization of frequency and time, and it is manifested by tiling the time-frequency plane with fixed uncertainty regions. For methods similar to and including the STFT, the resolution of time versus frequency is fixed throughout the entire time-frequency plane.

One of the initial benefits of wavelet signal decomposition is the dilation of the time windows. This time window dilation provides variable time-frequency trade-offs in the time-frequency plane. These trade-offs are illustrated by the right-hand side of Figure 2.1.

The time-frequency plane uncertainty tiling of the right-hand side of Figure 2.1 indicates that increased time resolution is achieved at higher frequencies, thereby allowing better resolution of quickly evolving time events that are composed of higher frequencies. Alternately, at lower frequencies, better frequency resolution is obtained at the cost of reduced time resolution, to better capture the slowly evolving events that are associated with the lower frequencies.

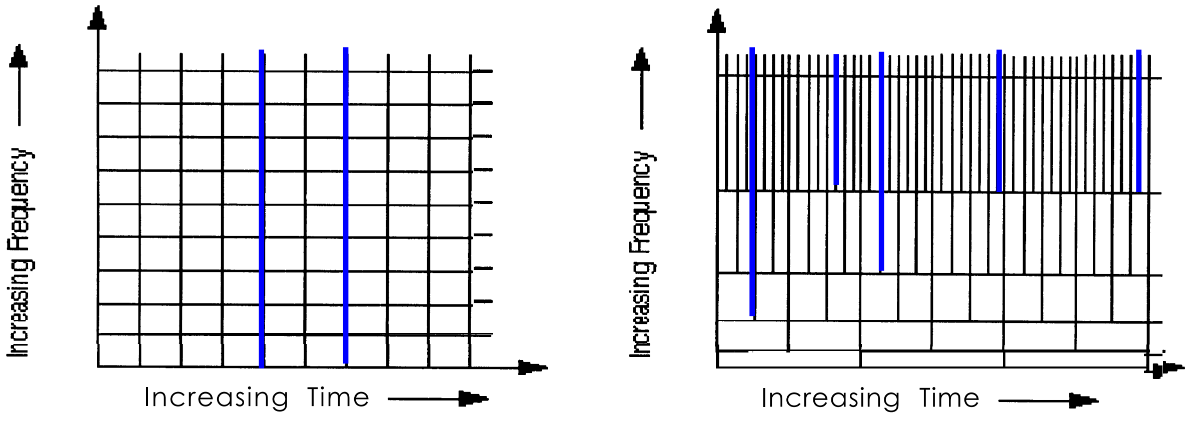


Figure 2.1 Typical Uncertainty Regions of a) STFT and b) Wavelet Decompositions

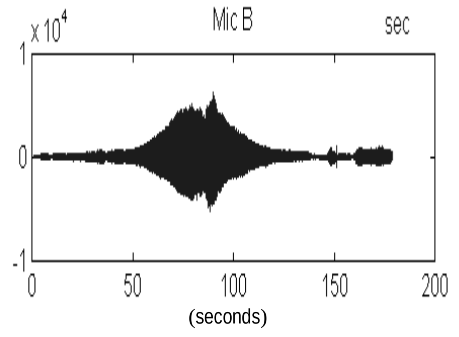
3. DEVELOPING AN ACOUSTIC ADAPTED WAVELET BASIS FUNCTION

An engineered CWT basis function has been developed for use in the vehicle classification problem, which will involve the processing of acoustic emissions. We have found that it leads to a viable, lower dimensional projection of the vehicle signal, resulting in a robust/unique characterization methodology for acoustic energy from vehicles.

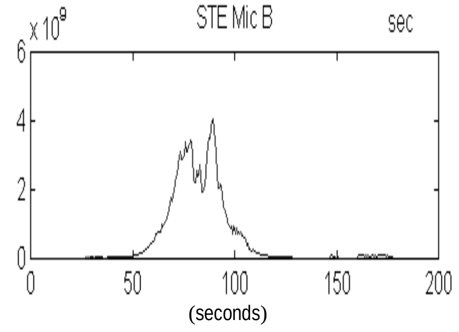
Based on following the formulation outlined in Equations (2.1)-(2.9), a number of candidate basis functions where developed. The main criteria for the selection of the basis function developed were: 1) readily interpretable ground vehicle CWT projections in the time/scale plane, 2) no mathematical impediment to using the basis function in real-time processing applications and 3) reduction of the basis functions to a compact, efficient implementation deployable on PC-based DSP platforms. The particular wavelet family developed has semi-infinite support in the time domain and can be modeled using causal real-rational transfer functions. This particular aspect of the basis function achieves the goal of a basis function capable of supporting real-time processing operations. In addition, the CWT basis function can be efficiently implemented using auto-regressive moving average (ARMA) techniques. The next section illustrates this basis function's ability to decompose the ground vehicle acoustic emissions in the time-scale plane.

4. CWT DECOMPOSITION OF GROUND VEHICLE ACOUSTIC EMISSIONS

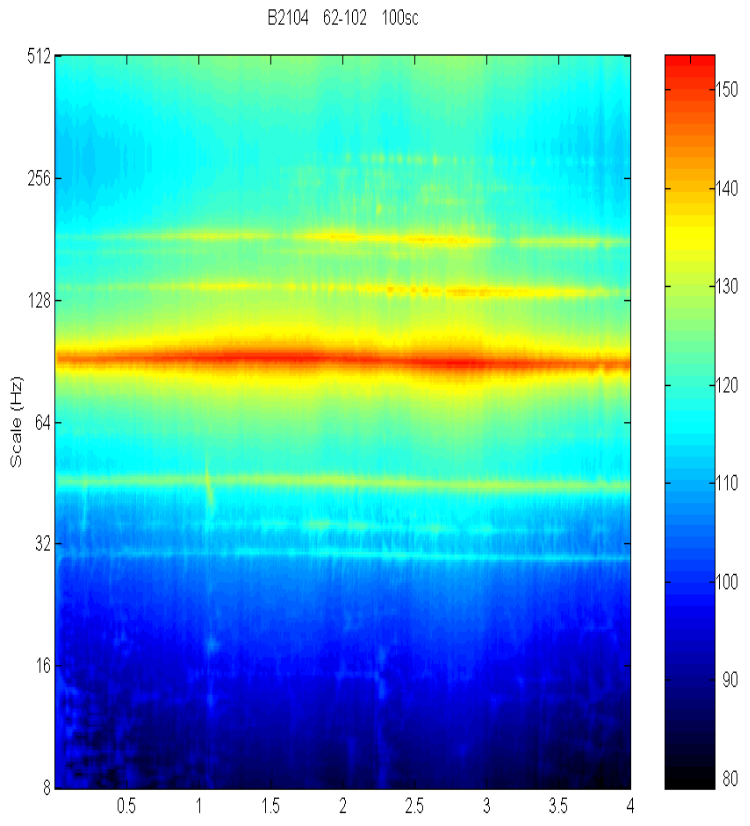
Figure 4.1 gives an example of using the CWT basis function to decompose a ground vehicle acoustic signal emission. A set of wavelet basis functions is used to decompose the acoustic emission from a Tank Transporter going left to right across the sensor package, at a speed of 30 kph and a closest point of approach of 75 meters. The decomposition is from the signal recorded on the B channel acoustic sensor and contains 40 seconds of data occurring during the time period of 62 to 102 seconds of the recorded pass. The CWT coefficients were then computed using the basis function developed. The CWT wavelet filters were evenly distributed across the frequency bandwidth of 8 Hz to 512 Hz in an octave fashion. The CWT was computed for 40 second interval and the wavelet filter coefficients were decimated along the time axis and stored in a matrix. The magnitudes of the CWT coefficients were converted to dB (decibel) scale and color coded using a hue saturation scheme that maps the lowest magnitude values to black, the highest magnitude values to red, and all other magnitudes mapped accordingly to the colors between black and red. To the right of the CWT is the map indicating dB values mapped to the continuum of color. The resultant CWT image generated is given in Figure 4.1. To the left of the CWT is the original time-domain signal plot and below that plot is a plot of the short-time energy of the signal. The 40 seconds of the signal processed with the CWT is approximately located around the center of the short-time energy plot (i.e. occurring between the 62th to 102th second of the recorded vehicle pass).



Time Domain Plot of Tank Transporter Signal
(File: Atc_2104.ad)



Short-Time Energy Plot of Tank Transporter Signal
(File: Atc_2104.ad)



[Horizontal Axis Time (increments of 10 seconds) , Vertical Axis Scale (Hz)]

CWT Decomposition Tank Transporter Signal

Figure 4.1 CWT of Tank Transporter Along with Time-Domain Plot and Short-Time Energy Plot

What is readily apparent from the CWT image of Figure 4.1 is the ability of the developed wavelet basis function to decompose the acoustic energy into a series of narrow-band wavelet structures. A strong response is exhibited at approximately 92 Hz with its associated harmonic 184 Hz. Another strong structure is visible at approximately 134 Hz. In addition, there are identifiable narrow band wavelet-based structures at 46 Hz, 36 Hz and 30 Hz. Most of the structures are detectable throughout the entire time duration of 40 seconds. For this particular pass, the closest point of approach was 75 meters and is aligned with the peak of the short time energy curve (i.e. approximately at the center of the CWT in Figure 4.1). At a vehicle speed of 30 kph (8.33 m/sec) the wavelet structures on the edge of the CWT are the result of acoustic emissions emanating from approximately 183 meters from the sensor package.

The ability of the CWT to decompose the acoustic emissions into readily identifiable structures is a particularly good result since such structure is the result of isolating vibrational effects associated with rotational machinery. These are the sort of signatures most desirable to extract from the acoustic emission since they are directly related to the underlying mechanical operation of the vehicle. Development of a decomposition method which leads to identifiable, wavelet-based features, such as exhibited in Figure 4.1, holds the promise of being a robust set of extractable features over a wide range of operating environments since they result from vibratory response linked to specific ground vehicle machinery. Keying off features linked to the mechanical operation of the vehicles leads to a feature extraction methodology with the highest degree of repeatability across a number of operating environments and this of course leads to greater reliability and higher performance of the resultant vehicle identification system.

5. ROBUSTNESS OF VEHICLE CWT ACROSS VARYING TARGET PARAMETERS

An important characteristic of any signal projection methodology being used to develop a pattern extraction technique for signal identification is the ability of the projection methodology to generate similar patterns for the same vehicle type under varying operations. Figure 5.1 shows the very good results obtainable from this CWT projection by displaying four different CWT decompositions of tank transporters having different directions and closest point of approach (CPA) with respect to the acoustic sensors. From Figure 5.1 all Tank Transporter acoustic emissions result in very similar wavelet signatures which are persistent in scale an exhibit very similar shaping along the overall time-scale plane. The repeated structure occurs through out all four CWTs with a strong response at approximately 92 Hz, harmonic at 184 Hz, response at 134 Hz, response at 46 Hz and 30 Hz.

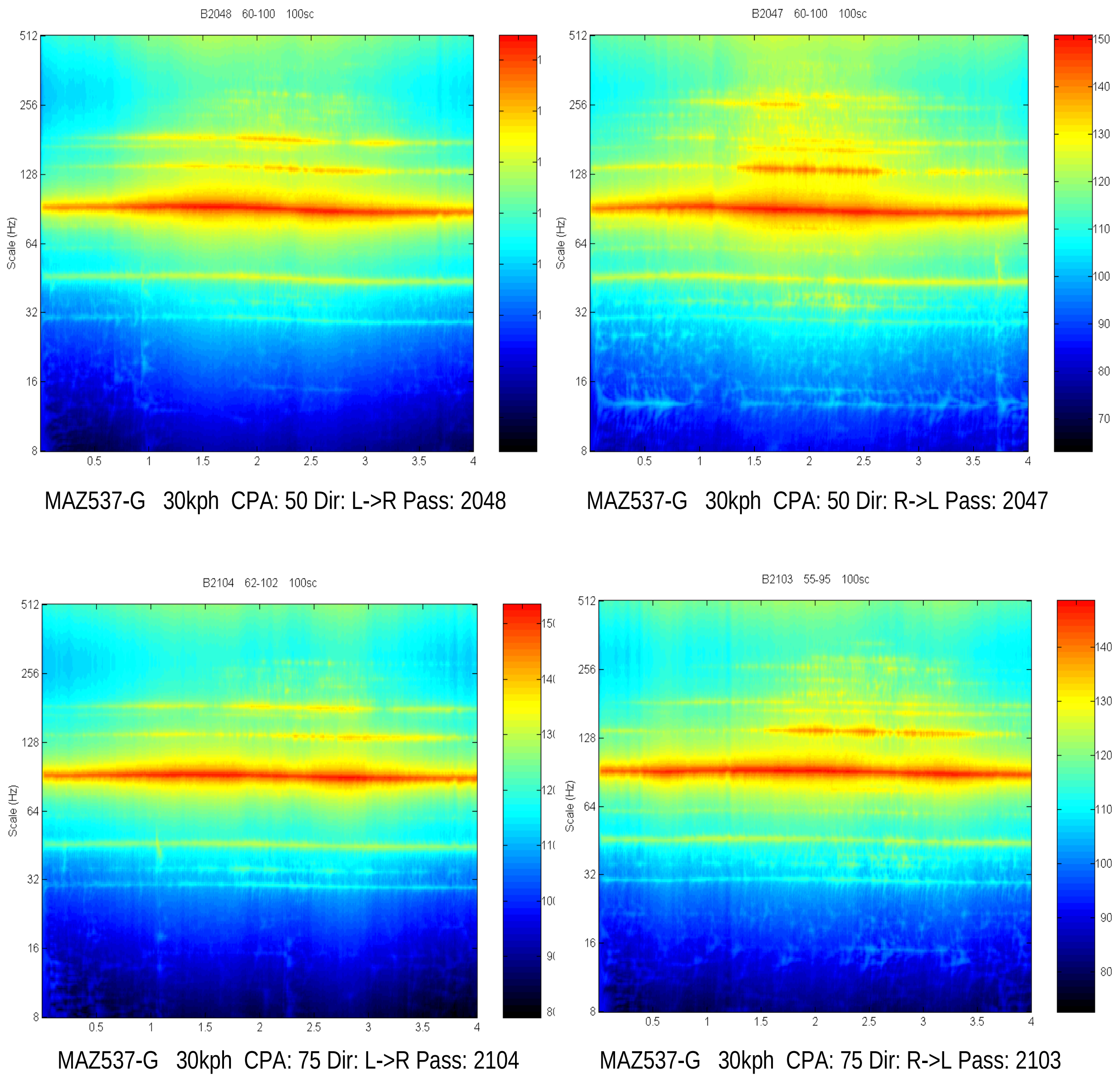


Figure 5.1 CWT From Tank Transporter Passes Having Different Directions and CPAs Relative to the Acoustic Sensors

6. UNIQUE CWT SIGNATURES FOR DIFFERENT TARGET TYPES

Having established in the previous section that the developed acoustic wavelet basis function produces a stable CWT projection for individual targets under a range of operational conditions such as direction and CPA relative to the sensor package, the next demonstration illustrates that this same signal projection technique leads to unique signatures for different target types. In order to develop a viable vehicle identification system, the features being extracted must provide separation in multidimensional feature space between the various target types in order for any sort of classification method to produce reliable vehicle identification. This implies that in order for the CWT to be effective for vehicle identification, it must generate a unique CWT projection relative to different targets in order to devise a wavelet-based extraction technique for use by a neural network pattern classification system.

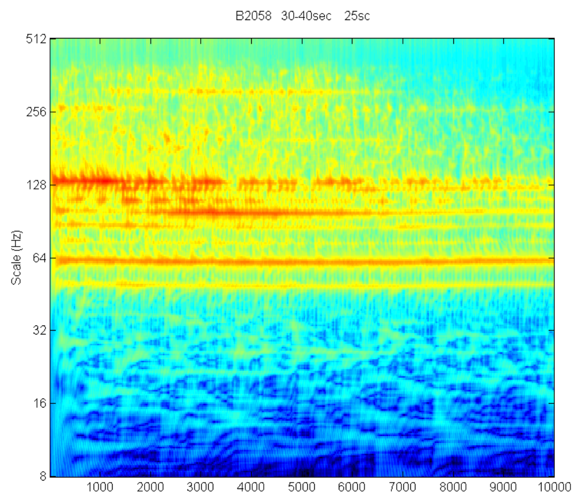
Figure 7.1 shows a side-by side comparison of CWTs for six different vehicle targets. Ten seconds of each target acoustic emission was used in order to highlight the unique CWT patterns existing for each of the separate vehicle targets. As is readily apparent to the eye, it is quite easy to detect that different targets lead to different CWT signatures.

7. NON-LINEAR CLASSIFICATION VIA NEURAL NETWORKS

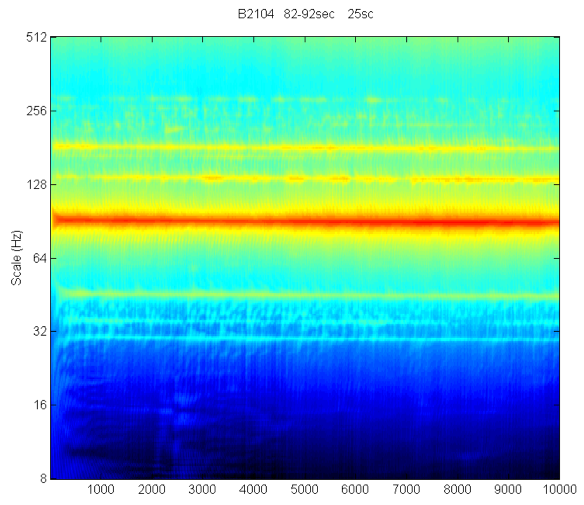
A natural choice for a vehicle classifier core that could process the wavelet-based vehicle signal feature vectors would be the old reliable Back-Propagation feedforward neural network. This classifier has been used in a wide variety of pattern classification applications over the last decade. Part of its success is attributed to the fact that neural networks scale, within reason, to address a large number of classes, can handle complex decision boundaries which are not necessarily linear, and have the ability to generalize to classes not previously seen but are within a given neighborhood of the data trained on. This last attribute allows the neural network to perform in a robust manner, across a wide range of data even in the presence of noise. Although, the BP is a bit of a workhorse with respect to pattern classification problems, it suffers from poor convergence (i.e. almost linear in the neighborhood of a local minimum) and long training times. There are many heuristics designed to speed up the convergence time, for example, the addition of a momentum term in the weight adjustment equation. However, this introduces yet another set of ad-hoc user assigned parameters (e.g. learning rate, momentum term, etc.) that leads to problems of repeatability and reliability of the training effort.

The main mathematical reason for the training performance woes exhibited by the BP algorithm deals precisely with the way the method treats the underlying nonlinear weight optimization problem. The training of a neural network can be view as a nonlinear optimization problem, that is, find a set of weights which minimize a global network error criterion. This criterion is usually an overall sum-squared error (SSE) of all the training pairs presented to the network. The BP algorithm is a computationally efficient form of a gradient descent nonlinear optimization routine. However, herein lies the seed of the training performance degradation. Gradient descent algorithms converge linearly in the neighborhood of local minimum. In order to achieve faster convergence (e.g. second order within the vicinity of a local minimum) and more reliable access to an error surface minimum without the need to set in an ad-hoc fashion a series of training parameters, one will need to apply more powerful nonlinear optimization methods. In this light, the basic BP algorithm with its reliance on gradient information makes for a very poor nonlinear optimization methodology. Essentially, any sophisticated numerical nonlinear optimization routine generally relies on at least second order information in conjunction with the gradient information to provide for faster convergence and more reliable accuracy.

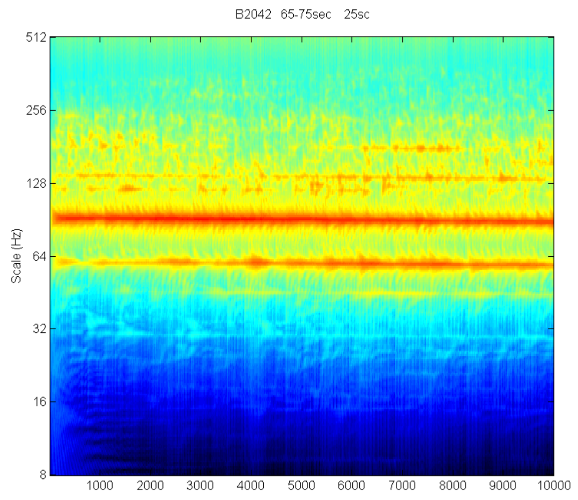
We have chosen to develop our neural network vehicle classifier cores using a well established nonlinear, second order, numerical optimization method known as the Levenberg-Marquardt [20] routine. The Levenberg-Marquardt routine is a modified version of a Gauss-Newton method with a trust region. The trust region effectively prevents the algorithm from diverging in a direction, which iteratively allows the error criterion to grow. This method results in faster training time and greater accuracy. It is not uncommon for it to achieve reduced training times and summed-squared errors which are an order of magnitude improvement over the conventional BP neural network algorithm. This method is fully explained in the following sections.



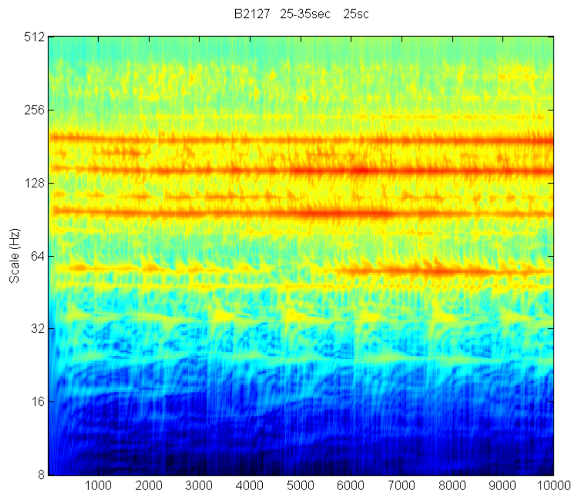
Lightweight Utility Vehicle (HMMWV)



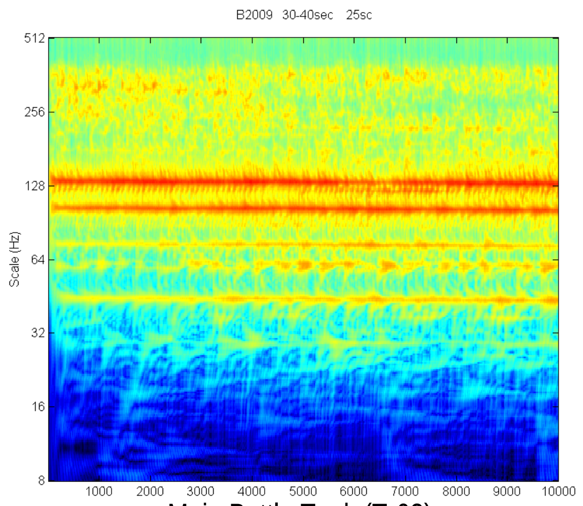
Tank Transporter (MAZ537-G)



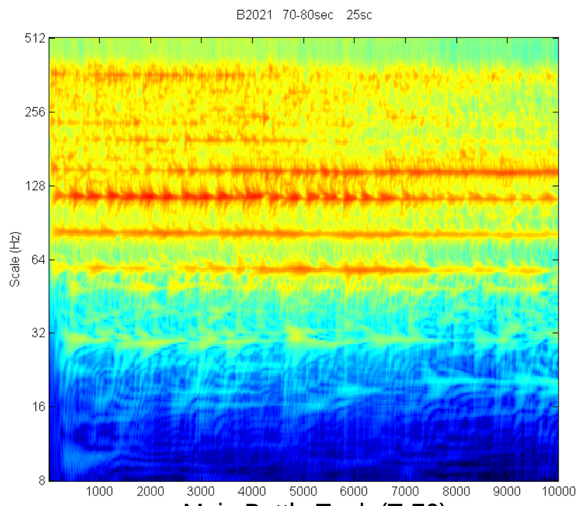
Tracked APC (BMP)



Main Battle Tank (M-60)



Main Battle Tank (T-62)



Main Battle Tank (T-72)

Figure 7.1 CWTs for Six Different Vehicle Types (time axis in milliseconds)

7.1. DERIVING THE LEVENBERG-MARQUARDT METHOD FROM A NEWTON APPROACH

Equation (7.1) gives an expression for the minimization of a nonlinear scalar function, which is dependent upon multiple parameters, expressed in the form of a vector. The search is conducted over the vector space whose dimension is equal to the number of elements in the parameter vector. A specific parameter vector represents just one point in the search space.

$$\min_{\hat{\mathbf{w}}} E(\hat{\mathbf{w}}) \quad where \quad \hat{\mathbf{w}} = [\hat{w}_1, \hat{w}_2, \cdots \hat{w}_v]^T \tag{7.1}$$

For nonlinear optimization an iterative technique known as Newton's method is employed to solve such minimization problems. The method consists of approximating the function $E(\hat{\mathbf{w}})$ about the current point $\hat{\mathbf{w}}(\tau)$ via a quadratic function. The quadratic formulation can then be minimized exactly and this is repeated at every iteration of the algorithm [20]. Expanding $E(\hat{\mathbf{w}})$ in a Taylor expansion about the point $\hat{\mathbf{w}}(\tau)$ we obtain the following approximation:

$$E(\hat{\mathbf{w}}) \cong E(\hat{\mathbf{w}}(\tau)) + (\hat{\mathbf{w}} - \hat{\mathbf{w}}(\tau))^T \nabla E(\hat{\mathbf{w}}(\tau)) + \frac{1}{2}(\hat{\mathbf{w}} - \hat{\mathbf{w}}(\tau))^T \nabla^2 E(\hat{\mathbf{w}}(\tau))(\hat{\mathbf{w}} - \hat{\mathbf{w}}(\tau)) \tag{7.2}$$

The iterative solution in a pure Newton form is:

$$\hat{\mathbf{w}}(\tau+1) = \hat{\mathbf{w}}(\tau) - \left[\nabla^2 E(\hat{\mathbf{w}}(\tau))\right]^{-1} \nabla E(\hat{\mathbf{w}}(\tau)) \tag{7.3}$$

Or equivalently the change in parameters after one iteration of the optimization algorithm is:

$$\Delta\hat{\mathbf{w}}(\tau) = -\left[\nabla^2 E(\hat{\mathbf{w}}(\tau))\right]^{-1} \nabla E(\hat{\mathbf{w}}(\tau)) \tag{7.4}$$

Lets examine a particular nonlinear scalar functional, which is in the form of a sum of squares.

$$E(\hat{\mathbf{w}}) = \frac{1}{2}\sum_{i=1}^{N} \varepsilon_i^2(\hat{\mathbf{w}}) \tag{7.5}$$

This functional in the form of a sum of squares is essentially equivalent to a SSE global error criterion for a neural network by considering the following relationships:

$$E(\hat{\mathbf{w}}) = \frac{1}{2}\sum_{i=1}^{N} \varepsilon_i^2(\hat{\mathbf{w}}) \equiv \frac{1}{2}\sum_{p=1}^{P}\sum_{k=1}^{m_L} e_k^2(p) = \frac{1}{2}\sum_{p=1}^{P}\sum_{k=1}^{m_L}\left(d_k^p - o_k(p)\right)^2 \tag{7.6}$$

There is a one to one relationship between the square error terms of form $\varepsilon_i^2(\hat{\mathbf{w}})$ and terms of the error outputs of the neural network: $e_k^2(p)$. The equivalence is just a simple accounting of indices that provides the mapping: $i \rightarrow (k, p)$. Where *k* represents a specific output of the neural network and *p* represents a specific training pair used to train the network. The aggregate total of error output terms across the entire training set are identical, that is $N = P \cdot m_L$, where *P* is the maximum number of training pairs used to train the network and m_L is the maximum number of outputs from the neural network. Also a side note is that the term $e_k^2(p)$ is also dependent on the neural network parameters (i.e. weights and biases) and that there also exists a one-to-one relationship between the elements of the vector $\hat{\mathbf{w}}$ and the weights and biases of the neural network. One way to express this relationship is as follows:

$$\hat{\mathbf{w}} = [\hat{w}_1 \quad \hat{w}_2 \quad \cdots \quad \hat{w}_V]^T = \left[w_{1,0}^{(1)}, w_{1,1}^{(1)} \quad \cdots \quad w_{m_1,m_0}^{(1)} \quad \cdots \quad w_{1,0}^{(L)}, w_{1,1}^{(L)} \quad \cdots \quad w_{m_2,m_1}^{(L)}\right]^T \tag{7.7}$$

Where the maximum number of elements in $\hat{\mathbf{w}}$ is equal to the maximum number of weights and biases in the neural network and can be calculated via the following formula:

$$V = \sum_{l=1}^{L} (m_{l-1} + 1) m_l \tag{7.8}$$

Where m_l represents the maximum number of neurons available in layer *l* of the network. The Hessian term, $\nabla^2 E(\hat{\mathbf{w}}(\tau))$, of Equation (7.4) contains second derivative terms of the form $\frac{\partial^2 E(\hat{\mathbf{w}})}{\partial \hat{w}_i \partial \hat{w}_j}$. Relationships (7.5)-

(7.8) illustrate the fact that the particular error criterion (i.e. nonlinear functional) we are looking to minimize for our feedforward neural network is in the form of a "sum of squares", hence there exists an approximation to the pure Newton optimization iteration of Equation (7.4) which can be performed but does not entail the computational cost of calculating second derivative terms. This method is know as the Gauss-Newton iteration and is derived in the following way.

An error vector of the individual error terms from Equation (7.5) can be expressed in the following way:

$$\xi\left(\hat{\mathbf{w}}\right)=\left[\varepsilon_1\left(\hat{\mathbf{w}}\right),\varepsilon\left(\hat{\mathbf{w}}\right)_2,\cdots\varepsilon_N\left(\hat{\mathbf{w}}\right)\right]^T \tag{7.9}$$

The Jacobian matrix is defined in the following manner:

$$\mathbf{J}(\hat{\mathbf{w}})=\begin{bmatrix} \frac{\partial\varepsilon_1(\hat{\mathbf{w}})}{\partial\hat{w}_1} & \frac{\partial\varepsilon_1(\hat{\mathbf{w}})}{\partial\hat{w}_2} & \cdots & \frac{\partial\varepsilon_1(\hat{\mathbf{w}})}{\partial\hat{w}_V} \\ \frac{\partial\varepsilon_2(\hat{\mathbf{w}})}{\partial\hat{w}_1} & \frac{\partial\varepsilon_2(\hat{\mathbf{w}})}{\partial\hat{w}_2} & \cdots & \frac{\partial\varepsilon_2(\hat{\mathbf{w}})}{\partial\hat{w}_V} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial\varepsilon_N(\hat{\mathbf{w}})}{\partial\hat{w}_1} & \frac{\partial\varepsilon_N(\hat{\mathbf{w}})}{\partial\hat{w}_2} & \cdots & \frac{\partial\varepsilon_N(\hat{\mathbf{w}})}{\partial\hat{w}_V} \end{bmatrix} \tag{7.10}$$

Each row of the Jacobian is effectively the gradient on the individual error with respect to the parameter vector $\hat{\mathbf{w}}$ and can be rewritten in the slight more compact notation:

$$\mathbf{J}(\hat{\mathbf{w}})=\begin{bmatrix} \nabla\varepsilon_1\left(\hat{\mathbf{w}}\right)^T \\ \nabla\varepsilon_2\left(\hat{\mathbf{w}}\right)^T \\ \vdots \\ \nabla\varepsilon_N\left(\hat{\mathbf{w}}\right)^T \end{bmatrix} \qquad \textit{where} \qquad \nabla\varepsilon_i\left(\hat{\mathbf{w}}\right)^T=\left[\frac{\partial\varepsilon_i\left(\hat{\mathbf{w}}\right)}{\partial\hat{w}_1},\frac{\partial\varepsilon_i\left(\hat{\mathbf{w}}\right)}{\partial\hat{w}_2}\cdots\frac{\partial\varepsilon_i\left(\hat{\mathbf{w}}\right)}{\partial\hat{w}_V}\right] \tag{7.11}$$

Repeated differentiation of Equation (7.5) gives the following:

$$\nabla E\left(\hat{\mathbf{w}}\right)=\mathbf{J}\left(\hat{\mathbf{w}}\right)^T\xi\left(\hat{\mathbf{w}}\right)=\sum_{i=1}^{N}\varepsilon_i\left(\hat{\mathbf{w}}\right)\nabla\varepsilon_i\left(\hat{\mathbf{w}}\right) \tag{7.12}$$

$$\nabla^2 E\left(\hat{\mathbf{w}}\right)=\mathbf{J}\left(\hat{\mathbf{w}}\right)^T\mathbf{J}\left(\hat{\mathbf{w}}\right)+\sum_{i=1}^{N}\varepsilon_i\left(\hat{\mathbf{w}}\right)\nabla^2\varepsilon_i\left(\hat{\mathbf{w}}\right) \tag{7.13}$$

For points near a minimum the Hessian, $\nabla^2 E\left(\hat{\mathbf{w}}\right)$, can be approximated as follows:

$$\nabla^2 E\left(\hat{\mathbf{w}}\right)\cong\mathbf{J}\left(\hat{\mathbf{w}}\right)^T\mathbf{J}\left(\hat{\mathbf{w}}\right) \tag{7.14}$$

Hence the need to calculate second order derivatives required by the term $\nabla^2\varepsilon_i\left(\hat{\mathbf{w}}\right)$ have been eliminated. This approximation is known as the Gauss-Newton algorithm and iteration rule becomes:

$$\Delta\hat{\mathbf{w}}=-\left[\mathbf{J}\left(\hat{\mathbf{w}}\right)^T\mathbf{J}\left(\hat{\mathbf{w}}\right)\right]^{-1}\mathbf{J}\left(\hat{\mathbf{w}}\right)^T\xi\left(\hat{\mathbf{w}}\right) \tag{7.15}$$

Unfortunately failure of the pure Gauss-Newton formulation will occur anytime the approximate Hessian matrix, $\mathbf{J}\left(\hat{\mathbf{w}}\right)^T\mathbf{J}\left(\hat{\mathbf{w}}\right)$ is singular (i.e. an inverse does not exists). For the approximate Hessian to be nonsingular, the Jacobians are required to have row rank *N*, which is not guaranteed to always be the case. The practical solution to this dilemma is to recognized that $\mathbf{J}\left(\hat{\mathbf{w}}\right)^T\mathbf{J}\left(\hat{\mathbf{w}}\right)$ is positive semi-definite and has a minimum possible eigenvalue of zero. Hence, augmenting this via the addition of an identity matrix scaled by any small, but numerically significant positive value will restore full rank and produce a matrix, which is nonsingular. This modification is known as the Levenberg-Marquardt (LM) algorithm and results in the following iteration rule:

$$\Delta\hat{\mathbf{w}}=-\left[\mathbf{J}\left(\hat{\mathbf{w}}\right)^T\mathbf{J}\left(\hat{\mathbf{w}}\right)+\mu\mathbf{I}\right]^{-1}\mathbf{J}\left(\hat{\mathbf{w}}\right)^T\xi\left(\hat{\mathbf{w}}\right) \tag{7.16}$$

The modification serves another purpose which turns out to be extremely useful when using Equation (7.16), it allows the establishment of a "trust" region when controlling the LM algorithm in software to train a neural network. What this means is that a training algorithm can be designed around Equation (7.16) such that the network weights are never updated unless the global error criterion, $E(\hat{\mathbf{w}})$ (see Equation (7.6)) decreases during the processing of the training set for any given specific training epoch (i.e. during any specific pass of the training set). To see how this control logic operates with respect to Equation (7.16), observe the two extremes of Equation (7.16).

$$For \quad \mu >> 1 \qquad \Delta\hat{\mathbf{w}} \approx -\mu^{-1}\mathbf{J}(\hat{\mathbf{w}})^T \xi(\hat{\mathbf{w}}) \equiv -\eta\nabla E(\hat{\mathbf{w}}) \qquad with \quad \mu^{-1} = \eta \tag{7.17}$$

$$For \quad \mu << 1 \qquad \Delta\hat{\mathbf{w}} \approx -\left[\mathbf{J}(\hat{\mathbf{w}})^T \mathbf{J}(\hat{\mathbf{w}})\right]^{-1} \mathbf{J}(\hat{\mathbf{w}})^T \xi(\hat{\mathbf{w}}) \tag{7.18}$$

For very large values of μ, the scaled Identity term dominates and the LM algorithm approaches a form of Gradient Descent with a very small step size as illustrated by Equation (7.17). For very small values of μ, the approximate Hessian term dominates and the LM algorithm approaches the Gauss-Newton algorithm (see Equation (7.15)). Our goal is to approximate the Gauss-Newton algorithm as much as possible, since the convergence of this algorithm in the vicinity of a minimum is second order. Therefore, when the algorithm is initially started a small value of μ is selected. For example, in the **Vehicle Monitor Simulation Environment (VMSE)** the initial value is set to $\mu = 0.001$, however the user is free to choose whatever initial value they would like. If the global error criterion decreases during the processing of one pass of the training data through the LM algorithm the value of μ is decreased for the next iteration (e.g. in the VMSE μ is decreased by a power of 10). However, if the error criterion does not increase, the weights of the network are not adjusted and the value of μ is increased (e.g. in the VMSE μ is increased by powers of 10) and the LM algorithm is recalculated. This continues until some value of μ is found which results in the error criterion decreasing. This is, in fact, the implementation of the trust region mentioned earlier. In the extreme case (i.e. μ must be increased to a very large value), the algorithm is forced to a gradient descent mode (see Equation (7.17)) which will eventually result in a decrease. This safety value is necessary since the Hessian used in the LM method is only an approximate Hessian, hence a check must be put in place in the event this approximation would attempt to drive the iteration in a direction of increasing error.

8. SOFTWARE IMPLEMENTATION: VEHICLE MONITOR SIMULATION ENVIRONMENT (VMSE)

Based on explanations of the previous section it is readily apparent that the development of a viable vehicle classifier scheme can involve a number of degrees of freedom. An important key element is the ability to rapidly prototype various vehicle classifier systems and evaluate their performance. Additionally, this prototyping environment should naturally provide a very good user interface for configuration and off-line training of classifier cores, which could then be exported to real-time, PC-based, combat vehicle monitoring systems. We accomplished this through the development of an object-oriented Vehicle Monitor Simulation Environment (VMSE). This section highlights key design details and implementation of this software environment.

8.1. UNDERPINNINGS OF THE VMSE OBJECT-ORIENTED DESIGN

In designing the VMSE, quite separate from the design concerns involving the user interface, were the concerns centering on developing a set of objects which could accommodate the current vehicle feature extraction methods along with any refinements / additions which may occur in the future. In concert with that objective was the need to construct a series of objects which could contain the various desired neural network cores while simultaneously 1) presenting a set of interfaces that are uniform among the various neural network cores, 2) seamlessly interacting and associating with the constructed vehicle signal feature objects and 3) operating in a manner which hides most of the complexity of the underlying operation from the user while still providing a fair amount of user control in terms of the configuration of the overall vehicle classifier. The following two subsections detail how these challenges were met via the development of the "Feature Group Object" and a comprehensive set of "Derived Neural Network Object Cores".

8.1.1. Feature Group Object Design

Figure 8.1 provides a conceptual view of the object classes constructed to house feature vectors, methods used to derive them and local / group property storage. The key design decision in constructing the Feature Group Class was the practical reality of providing for a way of housing all the extracted features from a group of signal files which would form the basis of a complete training set or a test set for vehicle classifications. The architecture

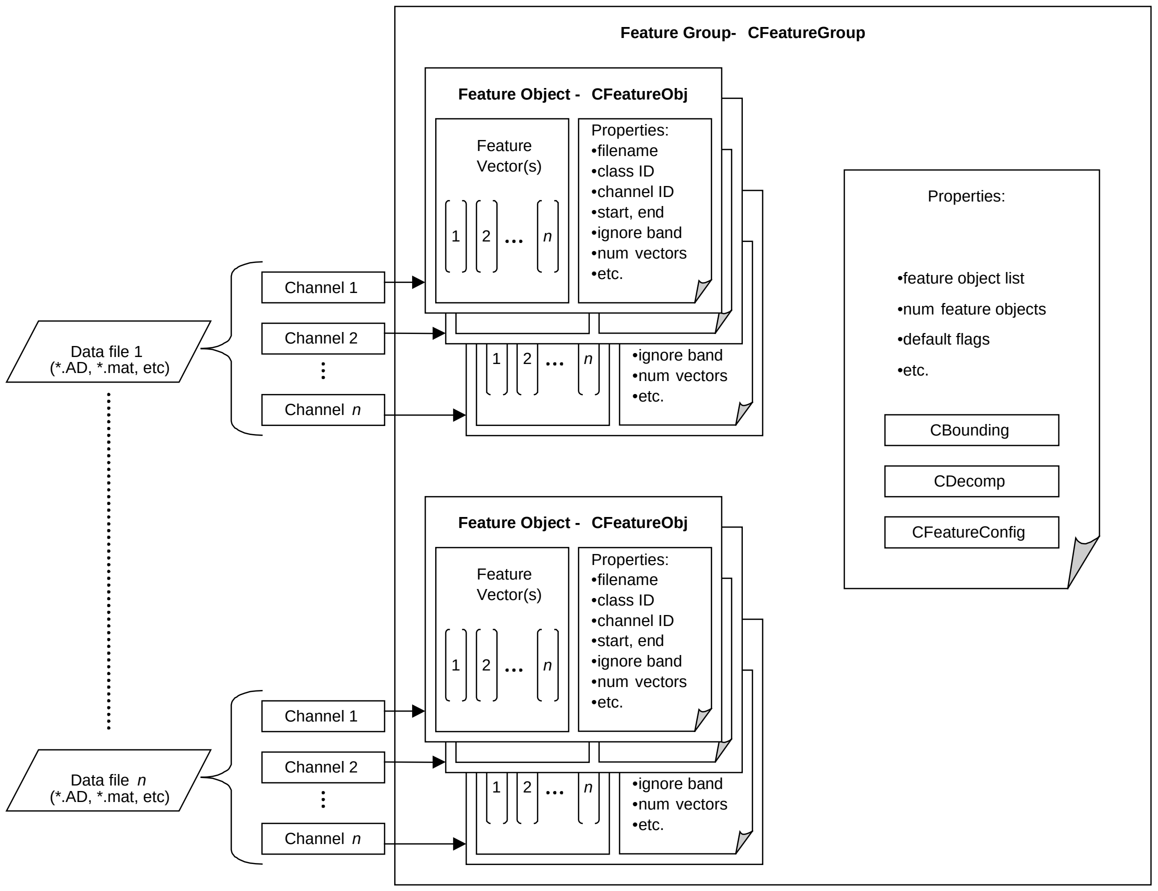
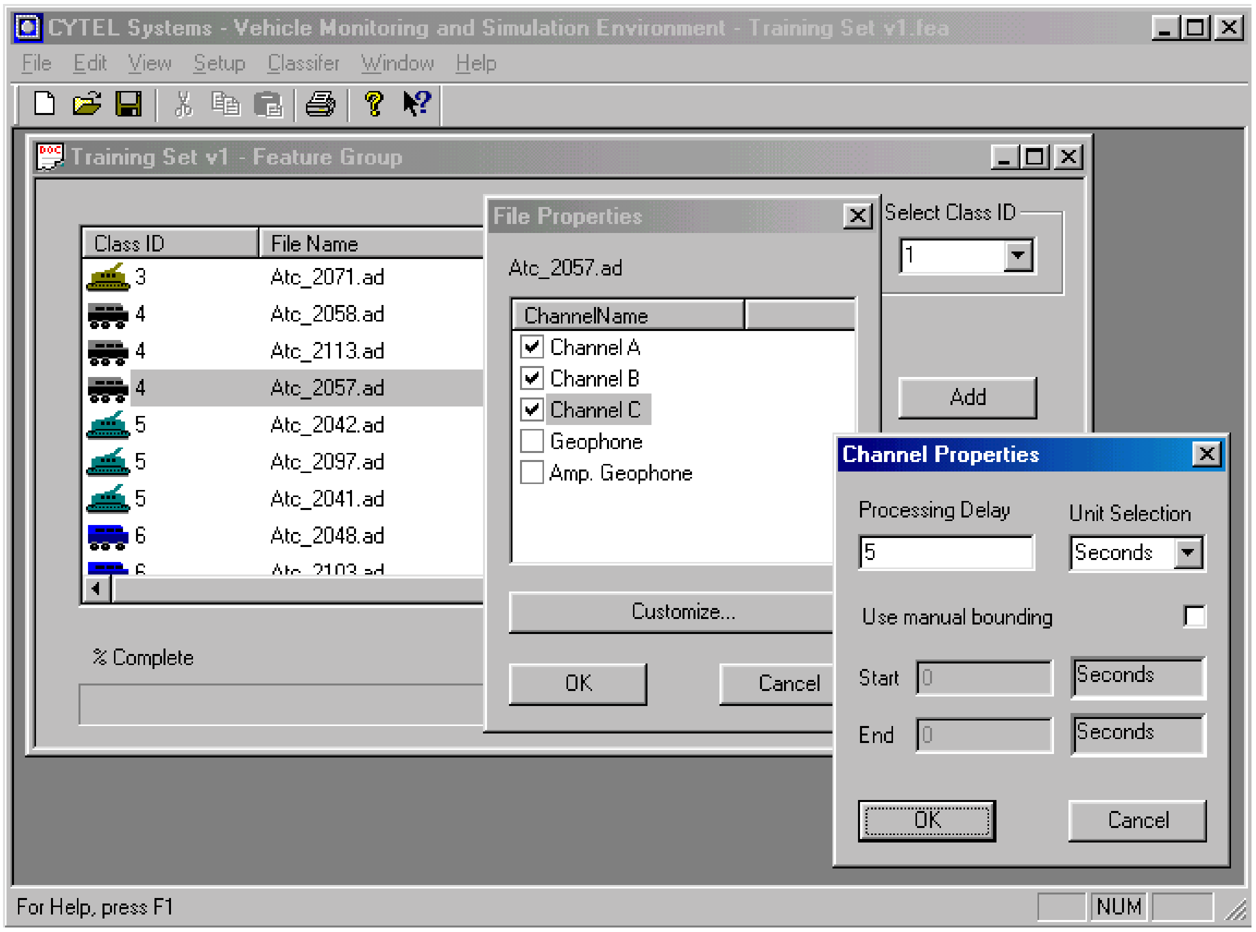


Figure 8.1 Feature Group Object Architecture

displayed in Figure 8.1 provides a means of handling such a situation. Figure 8.1 diagrams an architecture that supports extraction from a given vehicle signal file containing many channels of data. Each channel that is selected for feature extraction results in its own sub-object within the larger Feature Group Class that contains the storage for the features extracted and associated properties. Properties at the individual Feature object level capture information such as source of the original data, user assigned class ID (used eventually to determine the performance of the classifier), user assigned bounding options for individual channels, initial ignore bands on processing the signal channel, number of feature vectors created per channel, etc.

The overall Feature Group class contains additional information concerning 1) the specific global bounding options selected for all signal data in the feature group and 2) the global feature exaction methodology applied along with the specific parameters used. Containing this Feature Group Class information along with the original list control of signal files used to construct this class provides all the information necessary for this object to be 1) manipulated by Neural Network objects to achieve vehicle classification and 2) sufficient information such that a serialized and stored version of the Feature Group Object can be loaded at some future date into the VMSE and permit continued processing or updating.

Figure 8.2 shows an example of the user interface used to input data and make configurations all the way down to the vehicle signal channel level. Figure 8.2 illustrates the way individual signal files, possibly containing many vehicle signal channels, can be insert into feature group objects. Wavelet-based features extracted from the groups can be used either for training vehicle classifiers or for testing the ability of the vehicle classifier to perform vehicle identification. Global operations for performing vehicle signal segmentation, bounding and Wavelet-based feature extraction computations are available under the “Setup” menu of the VMSE mainframe. Due to the object based design of the environment these Feature Groups and settings can be stored and restored to the environment at the users convenience.


CYTEL Systems - Vehicle Monitoring and Simulation Environment - Training Set v1.fea

File Edit View Setup Classifer Window Help

Training Set v1 - Feature Group

Class ID	File Name
3	Atc_2071.ad
4	Atc_2058.ad
4	Atc_2113.ad
4	Atc_2057.ad
5	Atc_2042.ad
5	Atc_2097.ad
5	Atc_2041.ad
6	Atc_2048.ad
6	Atc_2103.ad

% Complete

File Properties

Atc_2057.ad

ChannelName	
✓ Channel A	
✓ Channel B	
✓ Channel C	
Geophone	
Amp. Geophone	

Select Class ID: 1

Channel Properties

Processing Delay: 5 Unit Selection: Seconds

Use manual bounding: ☐

Start: 0 Seconds

End: 0 Seconds

For Help, press F1 NUM

Figure 8.2 Formation of Feature Groups for Wavelet-based Feature Extraction and Channel Selections

8.1.2. Neural Network Classifier Object Design

Figure 8.3 shows a block diagram of how the neural network object cores connect to the Feature Group object to produce a vehicle classification function. Having a set of distinct classes which house these different types of data objects (i.e. feature objects verses neural network classifier objects) provides a significant degree of flexibility. For example, a Feature Group Object can serve as the training input or test data input to a number of different configured neural network vehicle classifiers. Under the VMSE a user function exists which facilitates a cut and paste operation to associate any configured neural network classifier in the environment to a user selected Feature Group Object.

Not explicitly shown in Figure 8.3, the neural network object cores are a derived set of objects. The hierarchy of the classes follows a logical progression. The base class consists of the basic neural network operations and storage, primarily containing all the elements necessary for a feedforward neural network class. The first derived class is the Back-Propagation (BP) neural network, which inherits all the properties of the base class and contains the data storage and local methods to perform the BP algorithm. The second derived class of neural network object cores is the Levenberg-Marquardt (LM) neural networks which inherits all the properties of the base and BP neural network class and provides the machinery to perform the LM training algorithm. Of course, to the user, all this machinery is invisible and they simply make the election, as shown in Figure 8.3, as to the type of neural network classifier core that they would like (i.e. BP or LM).

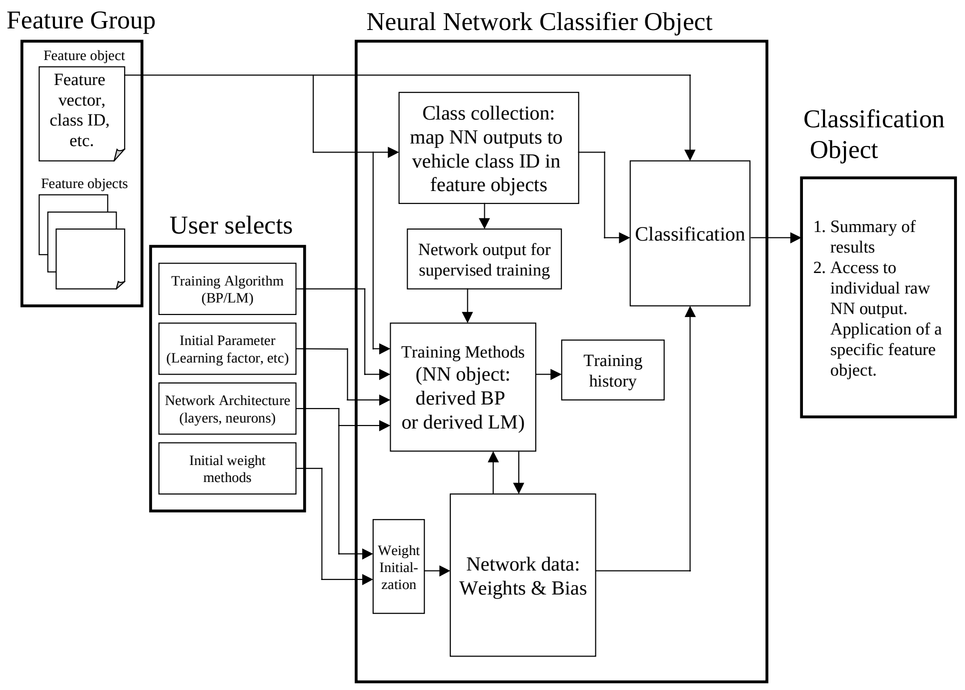
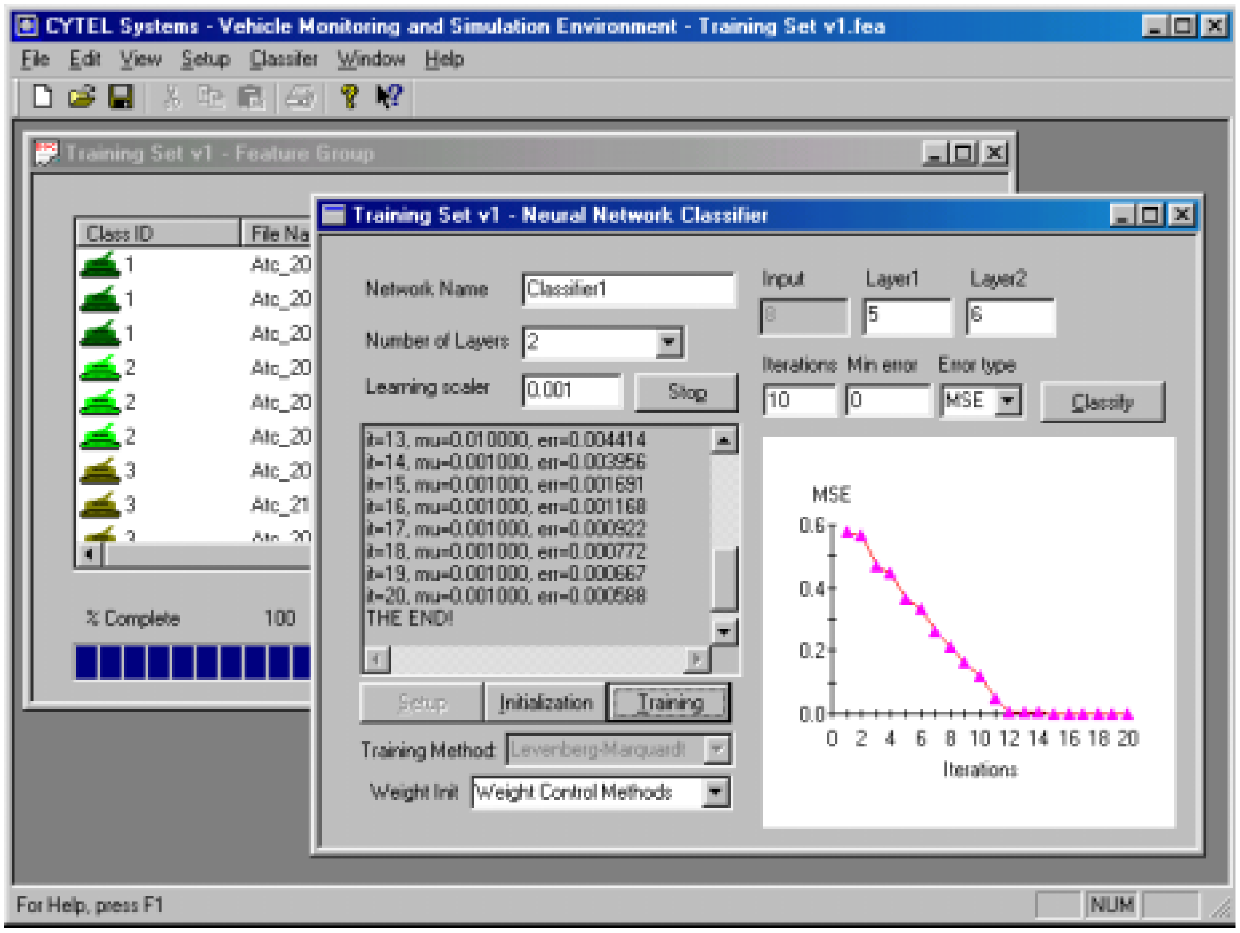

Feature Group
Feature object
Feature vector, class ID, etc.
Feature objects
Neural Network Classifier Object
Class collection: map NN outputs to vehicle class ID in feature objects
Network output for supervised training
Classification
Classification Object
1. Summary of results
2. Access to individual raw NN output. Application of a specific feature object.
User selects
Training Algorithm (BP/LM)
Initial Parameter (Learning factor, etc)
Network Architecture (layers, neurons)
Initial weight methods
Training Methods (NN object: derived BP or derived LM)
Training history
Weight Initial-zation
Network data: Weights & Bias

Figure 8.3 Interaction of Neural Network Classifier Objects with Feature Group Objects


CYTEL Systems - Vehicle Monitoring and Simulation Environment - Training Set v1.fea
File Edit View Setup Classifer Window Help
Training Set v1 - Feature Group
Class ID
File Na
Training Set v1 - Neural Network Classifier
Stop
Classify
Setup
Initialization
Training
MSE
0.6
0.4
0.2
0.0
0 2 4 6 8 10 12 14 16 18 20
Iterations
For Help, press F1
NUM

Class ID	File Na
1	Atc_20
1	Atc_20
1	Atc_20
2	Atc_20
2	Atc_20
2	Atc_20
3	Atc_20
3	Atc_21

% Complete 100

Network Name: Classifier1
Number of Layers: 2
Learning scaler: 0.001
Input: 8 Layer1: 5 Layer2: 6
Iterations: 10 Min error: 0 Error type: MSE

```
it=13, mu=0.010000, err=0.004414
it=14, mu=0.001000, err=0.003956
it=15, mu=0.001000, err=0.001691
it=16, mu=0.001000, err=0.001168
it=17, mu=0.001000, err=0.000922
it=18, mu=0.001000, err=0.000772
it=19, mu=0.001000, err=0.000667
it=20, mu=0.001000, err=0.000588
THE END!
```

Training Method: Levenberg-Marquardt
Weight Init: Weight Control Methods

Figure 8.4 Example of Training an LM Neural Network Vehicle Classifier on Extracted Vehicle Features

The neural network classifier object, as illustrated in Figure 8.3, takes feature vector data from individual Feature objects in the Feature Group. In addition the neural network accesses the user assigned class ID associated with the feature data. This class ID is necessary for the neural network training in order to translate the individual class ID to an appropriate desired output vector used in supervised training. In addition, this class ID information is used to provide a drill down function in the classification summary results of the vehicle monitor simulation to assess the classifiers performance. The neural network object maintains the architecture selections, class collection map (i.e. map from a user defined set of class IDs to the associated actual outputs of the neural network which represent those specific classes), neural network weights and its training history.

The neural networks once trained can be associated with and stored with a feature group for future processing. They can be shared within the VMSE among the various active feature groups in the environment. And finally, their trained weights and bias can be exported out of the environment for use in other applications, the main one being trained networks for a real-time, on-line, vehicle monitoring operation. Figure 8.4 gives a screen shot of a trained LM neural network associated with a Feature Group extracted from a set of combat vehicles.

8.2. VEHICLE CLASSIFIER SIMULATIONS USING THE LM CLASSIFIER

The following table gives the listing of vehicle data used in the simulations.

Table 8.1 Vehicle Data Used In Simulations

Vehicle Type	Class Designation	Filename	Channels Used	CPA (meters)	Direction
Main Battle Tank (T-62)	Class 1	ATC_2009.AD	3 acoustic	50	R->L
		ATC_2010.AD	3 acoustic	50	L->R
		ATC_2081.AD	3 acoustic	75	R->L
		ATC_2082.AD	3 acoustic	75	L->R
Main Battle Tank (T-72)	Class 2	ATC_2021.AD	3 acoustic	50	R->L
		ATC_2022.AD	3 acoustic	50	L->R
		ATC_2085.AD	3 acoustic	75	R->L
		ATC_2086.AD	3 acoustic	75	L->R
Main Battle Tank (M-60)	Class 3	ATC_2071.AD	3 acoustic	50	R->L
		ATC_2072.AD	3 acoustic	50	L->R
		ATC_2127.AD	3 acoustic	75	R->L
		ATC_2128.AD	3 acoustic	75	L->R
Lightweight Utility Vehicle (HMMWV)	Class 4	ATC_2057.AD	3 acoustic	50	R->L
		ATC_2058.AD	3 acoustic	50	L->R
		ATC_2113.AD	3 acoustic	75	R->L
		ATC_2114.AD	3 acoustic	75	L->R
Tracked APC (BMP)	Class 5	ATC_2041.AD	3 acoustic	50	R->L
		ATC_2042.AD	3 acoustic	50	L->R
		ATC_2097.AD	3 acoustic	75	R->L
		ATC_2098.AD	3 acoustic	75	L->R
Tank Transporter (MAZ537-G)	Class 6	ATC_2047.AD	3 acoustic	50	R->L
		ATC_2048.AD	3 acoustic	50	L->R
		ATC_2103.AD	3 acoustic	75	R->L
		ATC_2104.AD	3 acoustic	75	L->R

8.2.1. Parameter Selection for Feature Extraction and Neural Network Architecture in the Simulations

The wavelet-based feature vector computational parameters consisted of 1) 32 CWT filters octavely distributed across 32 to 512 Hz, 2) CWT coefficient values in dB, 3) CWT weighting process parameter of 97.5 ms and 4) partition into 4 zones.

The LM neural network classifier was used. It consisted of 2 neuron layers (i.e. input layer, hidden layer of neurons and output layer of neurons). Three of the vehicle files from each class were used to train the neural network classifier and verify the training performance. One set of “unseen” data, the forth file from each class, was held back and used to provide a test set of vehicle data not processed during training by the classifier to validate the vehicle monitor performance. Various weight initialization schemes were tested along with stopping criteria in order to determine training convergence performance and classifier core robustness to over training on the vehicle data set.

The resultant classifications from the simulations will be displayed in a confusion matrix style. Figure 8.5 gives a typical example. The numerical class codes (i.e. 1,2…6) are related to the respective vehicle types (i.e. T-62, T-72, HMMWV, … MAZ537-G) via the assignments specified in Table 8.1. The columns of the confusion matrix correspond to the type of vehicle data presented to the classifier; the rows indicate the type of vehicle classification declared by the classifier. If a number is on a diagonal position, it corresponds to the case where that number of presentations, of a particular vehicle type, are classified correctly. For example in Figure 8.5, at position (1,1) in the confusion matrix is the number 9. That number implies that 9 presentations of the vehicle type associated with Class 1 were classified correctly. Off diagonal numbers refer to mistakes made by the classifier. For example, in Figure 8.5 at position (5,3) in the confusion matrix is the number 2. That number implies those 2 presentations of the vehicle type associated with Class 3 were incorrectly classified as being Class 5.

Declared Classes	Classes Presented To Classifier					
	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Class 1	9	0	0	0	0	0
Class 2	0	9	0	1	0	0
Class 3	0	0	7	0	0	0
Class 4	0	0	0	8	0	0
Class 5	0	0	2	0	9	0
Class 6	0	0	0	0	0	9

Figure 8.5 Example of Confusion Matrix Used to Tabulate Vehicle Classification Results

8.2.2. Simulation 1

Number of neurons in the hidden layer is five. The weights were initialized using a fixed seed normal distribution random number. The target SSE criterion was 10.8 and the target iteration criterion was set to 50. The goal in this simulation is to determine the performance using a rather large SSE to effect a “light” training condition and to see the resultant performance using the random weight initialization. The final SSE error achieved was 8.76656 and this occurred on the 11th iteration. Table 8.2 and Table 8.3 gives the tabulated classification results.

Table 8.2 Tabulated Training Results for Simulation 1

Declared Classes	Classes Presented To Classifier					
	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Class 1	9	0	0	0	0	0
Class 2	0	9	0	0	0	0
Class 3	0	0	9	0	0	0
Class 4	0	0	0	7	0	0
Class 5	0	0	0	2	9	0
Class 6	0	0	0	0	0	9

Table 8.3 Tabulated Test Results for Simulation 1

Declared Classes	Classes Presented To Classifier					
	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Class 1	3	1	0	0	0	0
Class 2	0	2	0	0	0	0
Class 3	0	0	3	0	0	0
Class 4	0	0	0	3	0	0
Class 5	0	0	0	0	3	0
Class 6	0	0	0	0	0	3

The performance is as follows: 1) 3.7% of the data is misclassified when the system operates on the training data and 2) 5.56% of the data is misclassified when the system operates on the test data. This is a not a bad result given that the vehicle classifier is far from being fully trained (i.e. target error criterion was set high forcing a "lightly" trained situation). What will be observed in the next simulation is that only a few more training iterations are required to produce flawless results on this data set.

8.2.3. Simulation 2

Number of neurons in the hidden layer is five. The weights were initialized using a fixed seed normal distribution random number. The target SSE criterion was 5.4 and the target iteration criterion was set to 50. This simulation shows that with just a few more training iterations the performance observed in Simulation 1 is dramatically altered. The final SSE error achieved was 3.98499 and this occurred on the 13th iteration. Table 8.4 and Table 8.5 gives the tabulated classification results.

Table 8.4 Tabulated Training Results for Simulation 2

Declared Classes	Classes Presented To Classifier					
	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Class 1	9	0	0	0	0	0
Class 2	0	9	0	0	0	0
Class 3	0	0	9	0	0	0
Class 4	0	0	0	9	0	0
Class 5	0	0	0	0	9	0
Class 6	0	0	0	0	0	9

Table 8.5 Tabulated Test Results for Simulation 2

Declared Classes	Classes Presented To Classifier					
	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Class 1	3	0	0	0	0	0
Class 2	0	3	0	0	0	0
Class 3	0	0	3	0	0	0
Class 4	0	0	0	3	0	0
Class 5	0	0	0	0	3	0
Class 6	0	0	0	0	0	3

With just two more iterations than Simulation 1, the performance results of Simulation 2 give correct classification for every category.

8.2.4. Simulation 3

Number of neurons in the hidden layer is five. The weights were initialized using a fixed seed normal distribution random number. The target SSE criterion was 0.0054 and the target iteration criterion was set to 50. The goal in this simulation is to determine the classifier robustness to over training (i.e. the performance of the classifier on the test data set, if the classifier is over trained on the training data set). The over training condition is forced by setting a low SSE target-stopping criterion. The final SSE error achieved was 0.00360 and this occurred on the 20th iteration. Table 8.6 and Table 8.7 gives the tabulated classification results.

Table 8.6 Tabulated Training Results for Simulation 3

Declared Classes	Classes Presented To Classifier					
	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Class 1	9	0	0	0	0	0
Class 2	0	9	0	0	0	0
Class 3	0	0	9	0	0	0
Class 4	0	0	0	9	0	0
Class 5	0	0	0	0	9	0
Class 6	0	0	0	0	0	9

Table 8.7 Tabulated Test Results for Simulation 3

Declared Classes	Classes Presented To Classifier					
	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Class 1	3	0	0	0	0	0
Class 2	0	3	0	0	0	0
Class 3	0	0	3	0	0	0
Class 4	0	0	0	3	0	0
Class 5	0	0	0	0	3	0
Class 6	0	0	0	0	0	3

The flawless performance of the classification on the test data set suggests a level of robustness on the part of the LM vehicle classifier with respect to over training.

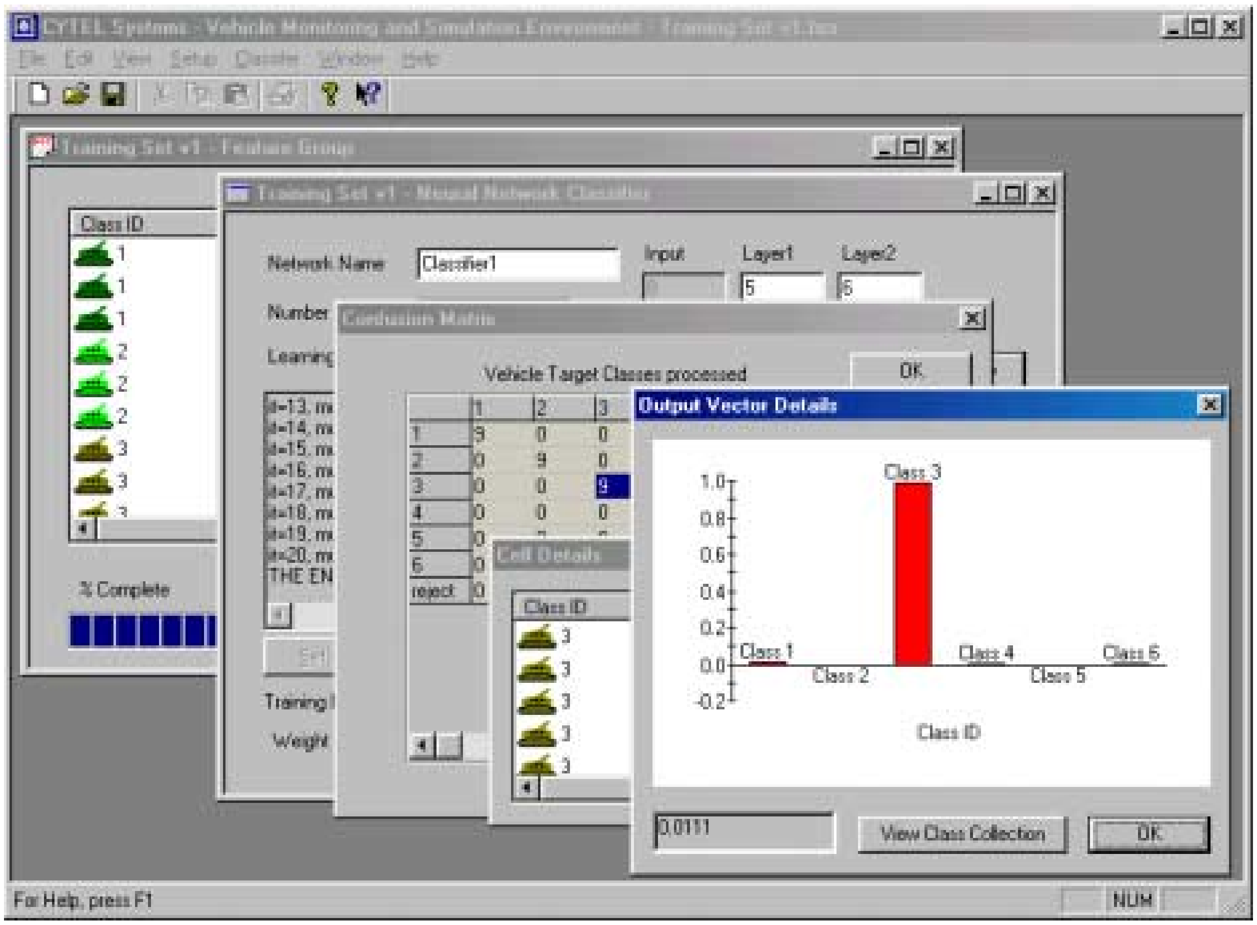
8.2.5. Simulation 4 through Simulation 6

The next three simulations are a repeat of the previous simulations 1-3 with the vehicle classifier weights being initialized and conditioned based on the statistics of the training set. The main conclusion with respect to the use of weight conditioning is that the LM vehicle classifier converges even faster than LM classifiers initialized with only random weights. Table 8.8 provides an aggregate overview of the performance results of Simulations 1 through 6. The acronym FSRN in Table 8.8 means: “Fixed Seed normal distribution Random Number” and WCM means “Weight Conditioning Method”.

Table 8.8 Summary Table for Simulation Results 1 to 6

Simulation	Initiali-zation method	Target Criteria		Final Training Metrics		Performance % Misclassification	
		Iteration	SSE	Iteration	SSE	Training	Testing
1	FSRN	50	10.8	11	8.7665	3.70	5.56
2	FSRN	50	5.4	13	3.9849	0	0
3	FSRN	50	0.0054	20	0.0036	0	0
4	WCM	50	10.8	8	8.6363	0	5.56
5	WCM	50	5.4	10	4.3564	0	0
6	WCM	50	0.0054	17	0.0006	0	0

An interesting observation is that the LM vehicle classifiers converged to an accurate vehicle classification scheme within a very small number of iterations. Typical Back-Propagation vehicle classifiers require at a minimum two orders of magnitude more in terms of final iteration values. In addition the LM vehicle classifier appears to provide outputs, prior to classifier thresholding which determines the class “declaration”, that are very strong and definitive. Figure 8.6 shows an example of a particular LM neural network raw output histogram. This histogram is generated dynamically by the user “clicking” on a cell of the “confusion matrix” in the VMSE. A list of vehicle signal channels responsible for the result emerges. “Clicking” on a given vehicle signal channel brings up the raw result from the vehicle classifier in response to the processing of the vehicle signal channel. One observes that the “winning” class declared by the LM neural network classifier, which is also the correct class by the way, is very dominate with respect to the other class values. This observation was a trend observed across the majority of vehicle classifications performed by the LM vehicle neural network classifier.



Network Name: Classifier1
Input:
Layer1: 5
Layer2: 6

	1	2	3
1	9	0	0
2	0	9	0
3	0	0	9
4	0	0	0
5	0		
6	0		
reject	0		

0.0111

Figure 8.6 Performance Drill Down to Show Neural Network Raw Output Due To Presentation of the Selected Vehicle Signal Channel Feature Vector

9. CONCLUSIONS

The simulations provide validation that the wavelet-based feature extraction and the LM neural network vehicle classifier provide a potent methodology for performing vehicle classification. Both the wavelet-based feature vectors and neural network architecture are very compact and this positively impacts the resources required to eventually perform real-time vehicle monitoring and classification. The simulations presented in the paper also illustrate the incredible quickness of the LM vehicle classifier to converge to a robust set of weights leading to high performance classification. This is directly attributable to the use of second order information for the nonlinear optimization of the neural network weights through the use of an effective modified-Gauss Newton training. For the standard BP algorithm, which relies on first order information though the use of gradient descent, training iteration cycles 1 to 2 orders of magnitude greater than that of the LM are generally required for successful training. In addition, the BP neural network suffers from repeatability issues in finding good solutions at the same accuracy achievable by the LM vehicle neural network classifier. Finally, the development of a sophisticated object based suite of software (i.e. the Vehicle Monitor Simulation Environment) has been instrumental in facilitating the quick prototyping, configuration and performance testing of a number of vehicle identification systems employing the wavelet-based/neural network methods.

10. ACKNOWLEDGEMENTS

This work was supported under the SBIR program of the U.S. ARMY ARDEC/TACOM, Picatinny Arsenal, NJ 07906-5000 under contract number DAAE30-98-C-1067.

11. REFERENCES

1. Coifman R, Meyer Y., and Wickerhauser MV, "Wavelet Analysis and Signal Processing", in Ruskai MB, Beylkin G., Coifman R., Daubechies I., Mallat S., Meyer Y., and Raphael L., eds., *Wavelets and their applications*, Jones and Bartett, Boston, 1992.
2. Chui CK, ed., *Wavelets: A Tutorial in Theory and Applications*, Academic Press, New York, 1992.
3. Beylkin G., Coifman R., and Rokhlin V., "Fast Wavelet Transforms and Numerical Algorithms", *Comm. Pure Appl. Math*. Vol. 44, pp. 141-183, 1991.
4. Newland, D.E., "Wavelet Analysis of Vibration, Part I: Theory", *J. Vib. Acoust., Trans. ASME*, Vol. 116, No. 4, pp. 409-416, Oct. 1994.
5. Daubechies, I., "The Wavelet Transform, Time-Frequency Localization and Signal Analysis," *IEEE Trans. Inform. Theory*, 36, 1990, 961-1005
6. Grossman, A., R. Kronland-Martinet, and J. Morlet, "Reading and Understanding Continuous Wavelet Transforms," in *Wavelets, Time-Frequency Methods and Phase Space*, J. Combes, et. al. (Eds.), Springer-Verlag, 1989.
7. Feichtinger H.G., and Grochenig K., "Gabor Wavelets and the Heisenberg Group: Gabor Expansions and Short-Time Fourier Transform From the Group Theoretical Point of View", in Chui CK, ed., *Wavelets: A Tutorial in Theory and Applications*, Academic Press, New York, 1992.
8. Tewfik A.H. and Hosur S., "Recent Progress in the Application of Wavelets in Surveillance Systems", *Proc. SPIE Conf. On Wavelet Applications*, 1994.
9. Rohrbaugh, R.A., "Application of Time-Frequency Analysis to Machinery Condition Assessment, *Proc. 27th Asilomar Conf. on Sigs., Syst., Comps.*, Vol. 2, pp. 1455-1458, 1993.
10. Rohrbaugh, R.A., Cohen, L., "Time-Frequency Analysis of a Cam Operated Pump", *Proc. of the 49th Mtg. of the Soc. for Machinery Failure Prevention Technology: Life Extension of Aging Machinery and Structures*, Virginia Beach, VA, pp. 349-361, April 18-20, 1995.
11. Yan, Tinghu, Zhong, "Artificial Neural Network Technique and Its Applications to Rotating Machinery Fault Diagnosis", *J. of Vib. Engrg.*, Vol. 6, pp. 205-212, 1993.
12. Mendel, J.M., *A Prelude to Neural Networks: Adaptive and Learning Systems*, PTR Prentice Hall, 1994.
13. Haykin, S., *Neural Networks: A Comprehensive Foundation*, MacMillan Press Ltd., 1994.
14. Teller, B.A., Harold H. Szu, Kiang, R.K., "Classifying Multispectral Data By Neural Networks", *Telematics and Informatics*, Vol. 10, No. 3, pp.209-222, 1993.
15. Freeman J.A., Skapura, D.M., Neural Networks Algorithms, Applications and Programming Techniques, Addison-Wesley Publishing Company, 1991.
16. Lopez, J.E., Yeldham, I.F., Oliver, K., Protz, M., "Hierarchical Neural Networks for Improved Fault Detection Using Multiple Sensors", *American Helicopter Society 52nd Annual Forum*, Washington, D.C., June 1996.
17. Lopez, J.E., "Performance of Wavelet/Neural Network Fault Detection Under Varying Operating Points", *Proceedings of the 66th Shock and Vibration Symposium*, Vol. 1, pp. 209-217, Oct.30 - Nov. 3, Biloxi, MS 1995.
18. Lopez, J.E., Oliver, K., "Improved Analysis Tools for Wavelet-Based Fault Detection", *IASTED International Conference, Signal and Image Processing - SIP- 95*, Las Vegas, NV, November 20-23, 1995.
19. Grimson, W.E., *Object Recognition By Computer*, The MIT Press Cambridge, MA, 1990.
20. Dennis, J.E., Schnabel, R.B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Englewood Cliffs, NJ, Prentice-Hall, 1983.